The discussion about the xvfat

2005.05.20

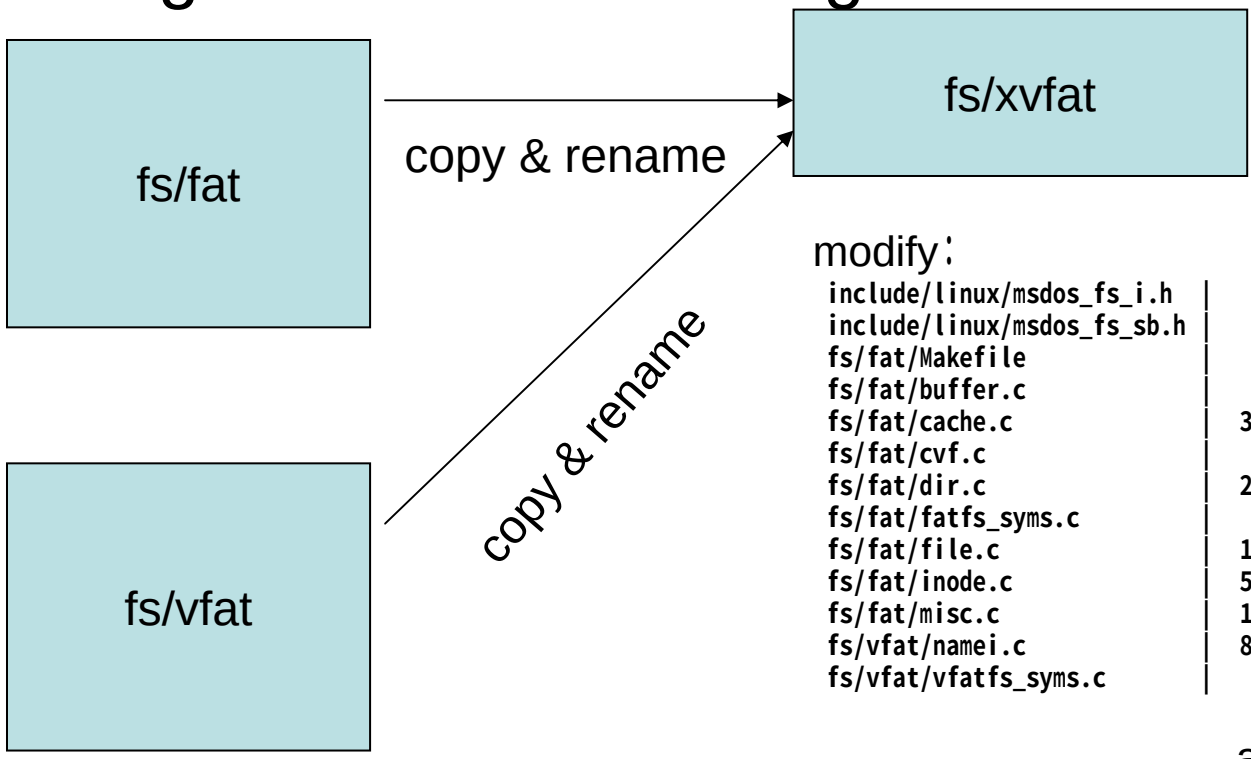
machida@sm.sony.co.jp

Agenda

- The introduction about the xvfat
- Regarding unplugged device/media
- Moving to Kernel 2.6
  - porting
  - Feedback to the community

Current source code organization

- designed to be coexisting with conventional FAT



modify:

```
include/linux/msdos_fs_i.h |    4
include/linux/msdos_fs_sb.h |   11
fs/fat/Makefile            |   12
fs/fat/buffer.c            |   39
fs/fat/cache.c             |  366 ++++++-
fs/fat/cvf.c               |   50 -
fs/fat/dir.c               |  228 +++-
fs/fat/fatfs_syms.c        |   41
fs/fat/file.c              |  170 +++
fs/fat/inode.c             |  562 +++++++++---
fs/fat/misc.c              |  129 +-
fs/vfat/namei.c            |  842 ++++++++++++++---
fs/vfat/vfatfs_syms.c      |   52 -
```

The elevator function modified,too (under drivers)

add:

```
2020 fs/xvfat/fwrq.c
1015 fs/xvfat/rdchk.c
 610 fs/xvfat/tz.c
  50 include/linux/rdchk.h
  41 include/linux/fat_tz.h
 224 include/linux/xvfat_fs.h
 245 include/linux/fwrq.h
```

Introduction about the xvfat

- See material shown at CELF tech meeting
  - 20050325-xvfat-intro-en.pdf

- Some information missed in the above material
  - new ioctl() for a file system
    - Add counter to track media status changes
      - Formerly just two state bits to be checked
      - However, this method can’t detect status change in some cases
  - Japanese support
    - →Include bug fixes related to unicode

Regarding unplugged device/media

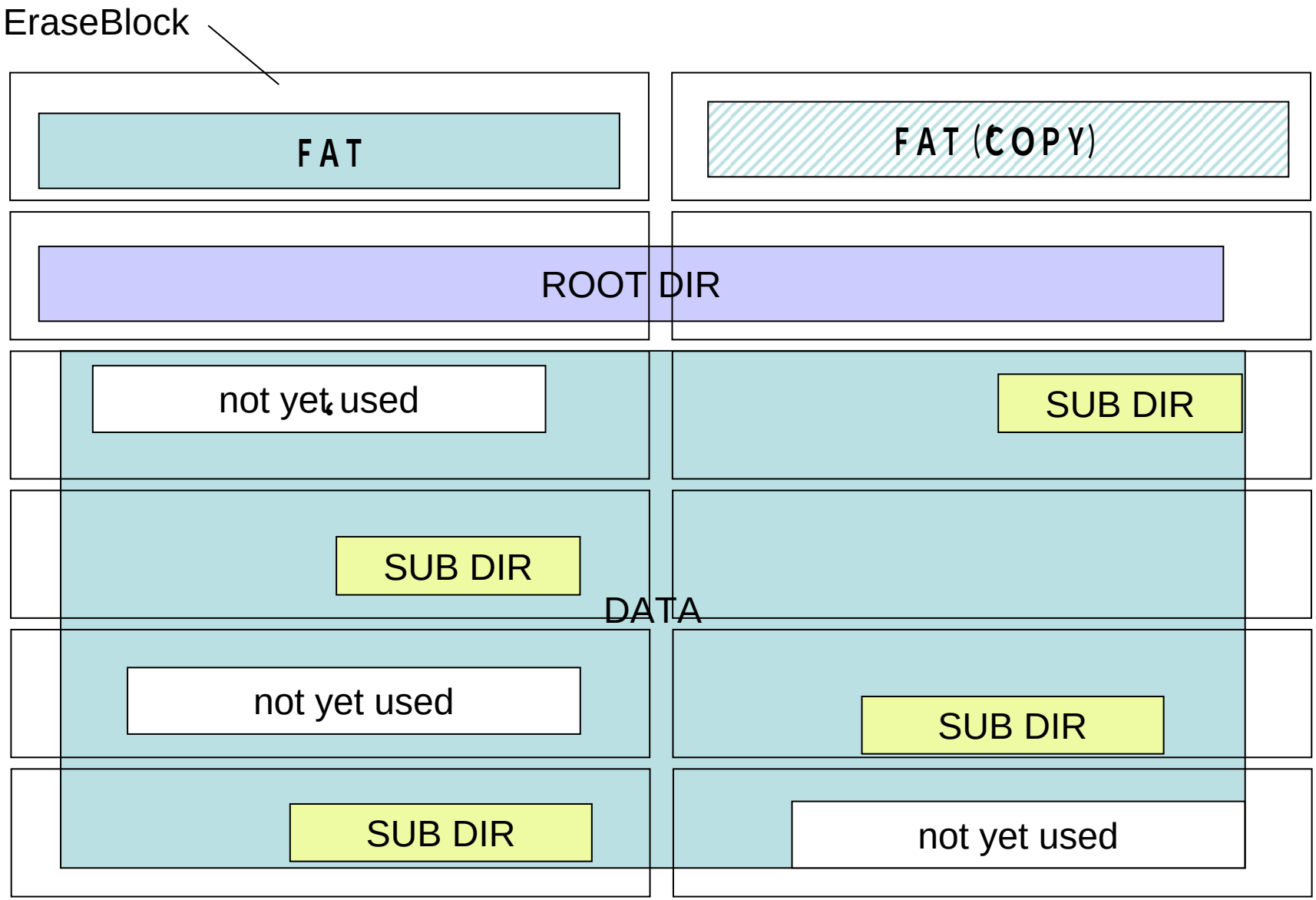
- The Detection & the Notification of unplugging
  - Media unplugging　Poll media status to detect unplugging
  - Device unplugging　Record device I/O error
    - After device unplugged, the system behavior is depended on device and it's driver

    →
    - It's not so robust, on device unplugged.

- The System call for device operations to un-plugged device/media always return error
  - Internal un-plug checking is taken place, at every device operations

- Preserve the order of written data (Reordering is not happened beyond EraseBlock)
- Synchronous writing
- Aggregating small data so that data will be stored in a single EraseBlock
  - It's important that FAT reside in EraseBlock
  - Transaction control utilizes property which EraseBlock can be written atomically and/or efficiently in most case.

FAT construction & EraseBlock



Example:create()

- Return error code ,if I/O error encountered
- Start the transaction
- Record adding entry to the directory
- Commit the transaction
- Write immediately updates on FAT (except for release)

example:write() to a file

- Return error code ,if I/O error encountered
- Start the transaction
- Write content of file
- Write immediately updates on directory and FAT
- Commit transaction

Moving to Kernel 2.6(1/3)

- Working with the community
  - Isolate the patches by each feature
  - Use existing mechanism inside kernel w or w/o modification
- First, isolate the current patches for kernel 2.4.20 by each feature
  - After that, investigate each feature
- Source code e organization
  - Utilize and share fs/fat, fs/vfat code, if possible

Moving to Kernel 2.6(2/3)

- Transaction control by EraseBlock unit
  - Implementation on 2.4.20
    - Original implementation
    - EraseBlock size is provided through MS parameter table in source code
  - Possible implementation on 2.6
    - Use journaling-API (jbd) **?**
      - Doc could be produced by Documentation/DocBook/journal-api.tmpl
    - Use RAM area as external journal to record transactions
    - Journaling FAT fs could be implemented using internal journaling
    - EraseBlock size

    → guess EraseBlock size from device parameters provided by driver.
      - See Documentation/block/biodoc.txt
      - Is max_seg_size suitable?

  - Cluster allocation also could take account EraseBlock?
    - For example, prefer to select EraseBlock which has a series of empty sectors from beginning…

Moving to Kernel 2.6(3/3)

- Detection and response of unplugging
  - Use HOTPLUG **?**
  - Kernel 2.6 add time out handler to functions in the ll_rw_block.c
    - It could be utilized unplugged device operations

- Control writing order
  - Implementation on 2.4.20
    - Reorder can be taken place inside in EraseBlock
    - own transaction control function
    - own in-order elevator function
  - Possible implementation on 2.6
    - enable /disable reordering in a EraseBlock
    - enable/disable taking account of head seek time
    - Specifying existing block device queue parameters can implement EraseBlock oriented operations, we need?
      - NO:
      - Block device queue can’t handle boundary of EraseBlock, correctly
      - Need to implement new elevator function

Target Product

- Embedded storage device
  - There are two path to access
    - Linux mounts embedded storage directly.
    - PC mounts embedded storage through USB (product act as USB client)
  - Flash Memory and so on
- Hot pluggable storage
  - Memory Card, CamCoder and DSC though USB
    - Linux mounts USB mass storage
    - product act as USB host
  - BUS connected storage
    - Memory card reader connected to PCMCIA card bus
- Need more consideration about HDD

Discussion(1/2)

- JBD can be used、maybe, but the modification would be needed
  - Data submitting timing control may be needed with own implementation
  - JFS may uses own kernel daemon, not using kjournald
- It is no easy to make developers understood why FAT is important
  - There is the possibility the discussion cannot be go on unless the goal and the assumption is not clear.
- The method “operations based on EraseBlock unit” is good ?
  - Methods which is writing WriteBlock unit, could be achieve good co-operation with under laying TransLayer?
  - Reading & writing by EraseBlock unit in could use burst mode operation on NAND flash

Discussion (2/2)

- Need cooperation with under laying TransLayer, if possible
  - Reading by Write Block unit could achieve good error recovery, than giving up using all data in a Erase block.
  - Using data copies inside in TransLayer (eg cache) could achieve good error recovery.
- Adding journaling function to FAT sounds good
  - We have to take care that the log does not consist with data about the media while removed, written by PC.
  - Need to more investigate about advantages - Is there the difference between embedded case

    and hot plug case ?
- Need to more investigate about difference between HDD support,USB Mass Storage support,and embedded flash memory support
  - seek time, existence of TransLayer, EraseBlock length、WriteBlock length, and so on